

Utilisation de la distance d'édition pour l'appariement sémantique de documents XML

Mai Dong Le*, Karen Pinel-Sauvagnat**

* ledongbk@yahoo.com

**IRIT-SIG

118 route de Narbonne
31 062 Toulouse Cedex 4
sauvagnat@irit.fr

Résumé. A notre connaissance, peu de travaux utilisent directement la théorie des graphes pour la recherche d'information structurée ad-hoc, c'est à dire pour calculer l'appariement approximatif entre une requête structurée et un document de type XML. Dans cet article, nous proposons une adaptation de l'algorithme de Tai (1979) pour le calcul de la distance d'édition entre l'arbre des documents et l'arbre de la requête. Cet algorithme est combiné à une recherche sur le contenu des documents.

1 Introduction

L'accès aux documents structurés de type XML (*eXtensible Markup Language*) soulève de nouvelles problématiques liées à la co-existence de l'information structurelle et de l'information de contenu. Les approches de l'état de l'art en Recherche d'Information Structurée considèrent que la dimension structurelle permet de mieux répondre aux différentes attentes des utilisateurs, en se focalisant sur leur besoin. Les méthodes proposées s'appliquent à renvoyer des parties de documents répondant de manière *spécifique* et *exhaustive* au besoin en information de l'utilisateur, que celui-ci soit formulé sous forme de requête structurée ou non. On trouvera dans Pinel-Sauvagnat et Boughanem (2006), Manning et al. (2008), Fuhr et al. (2008) et Geva et al. (2009) un état de l'art de la recherche d'information structurée actuelle.

De nombreuses approches utilisent la représentation arborescente des documents XML pour la recherche d'une information pertinente. Cependant, à notre connaissance, la théorie des arbres (et plus largement la théorie des graphes) n'est pas ou peu utilisée de façon explicite pour la recherche. Cette théorie est pourtant très puissante et pourrait donner des indications précieuses sur les algorithmes de recherche les plus intéressants à étudier.

Nous proposons dans cet article d'appliquer l'algorithme de Tai (1979) pour calculer la similarité de structure entre la requête et les documents. Cet algorithme, bien qu'assez simple, sert de base à la majorité des autres algorithmes d'appariement, et pourra donc servir de socle à de futures propositions. La mesure de similarité que nous proposons est la combinaison de cette similarité de structure avec une similarité de contenu, évaluée à l'aide d'un moteur de recherche plein texte classique.

Le reste de l'article est organisé comme suit: dans un premier temps (section 2), nous présentons ce qu'est l'appariement de graphes et d'arbres, ainsi que quelques applications à la recherche d'information structurée. Dans la section 3, nous proposons notre adaptation de l'algorithme de Tai (1979) à la recherche d'information structurée. La section 4 illustre cette adaptation sur un exemple.

2 Etat de l'art

2.1 Graphes et arbres

Les graphes sont une structure de données utile pour la représentation d'objets. Typiquement, des parties d'un objet complexe sont représentées par des *nœuds* (ou des sommets), et relations entre ces parties par des *arêtes*. Des *labels* (ou étiquettes) et attributs pour les nœuds et des arêtes sont employés pour incorporer les informations supplémentaires dans une représentation de graphes.

Si des graphes sont employés pour la représentation d'objets, trouver des objets similaires à d'autres revient à déterminer la similitude entre les graphes : on appelle cela *l'appariement de graphes*. Il y a deux types d'appariement de graphes (Bunke (2000)): l'appariement exact et l'appariement approximatif. L'appariement exact de graphes n'est pas très utile dans la recherche d'information, où on ne cherche pas à répondre de façon exacte au besoin de l'utilisateur. Nous nous intéressons plus à l'appariement approximatif, pour lequel les labels et le contenu d'un nœud peuvent jouer un rôle important.

L'appariement approximatif de graphes est une méthode basée sur le calcul de mesure de similarité de deux graphes. Il y a deux types principaux d'appariement approximatif : (1) *l'appariement basé sur le sous-graphe commun maximal* (Hidovi D., Pelillo M. (2004), Wallis et al. (2001), Bunke et Shearer (1998)) et (2) *l'appariement basé sur la distance d'édition* (Bunke (1998) (2000)).

1. Dans le premier cas, le sous-graphe commun maximal G de deux graphes G_1 et G_2 est un sous-graphe de G_1 et de G_2 ayant le nombre de nœuds maximal parmi tous ces sous-graphes. Plus deux graphes sont similaires, plus leur sous-graphe maximal est grand.
2. L'appariement basé sur la distance d'édition est une extension de la distance de chaînes de caractères dans le domaine des graphes. La distance d'édition de deux graphes G_1 et G_2 est définie comme le nombre minimum d'opérations nécessaires pour transformer G_1 en G_2 . Plus cette distance est petite, plus deux graphes sont similaires: cette distance convient donc pour mesurer la similarité de deux graphes.

Les graphes généraux ont cependant des structures et propriétés complexes, qui sont difficiles à utiliser dans la recherche d'information structurée. Dans notre cas, les documents XML, comme de nombreux autres objets, peuvent être représentés par des arbres, qui sont un cas particulier de graphes n'ayant aucun cycle.

Dans des applications basées sur des arbres, la similarité entre ces arbres est importante. Il y a trois types d'appariements principaux d'arbres : la *distance d'édition* (Tai (1979), Zhang (1996), Demaine et al. (2007), Sasha et Zhang (1997), Klein (1998)), la *distance d'alignement* (Jiang et al. (1995)) et l'*inclusion d'arbres* (Chen (1998)). On trouvera dans Lee (2009) le détail de ces algorithmes pour l'appariement. Nous proposons dans la section suivante quelques exemples d'applications de ces algorithmes à la recherche d'information structurée.

2.2 Applications à la recherche d'information structurée

Abdeslame Alilaouar (2007) utilise un type d'appariement basé sur la distance d'édition des arbres, pour l'interrogation flexible de données semi-structurées en général et des données XML en particulier. Les travaux redéfinissent un cadre permettant de déterminer jusqu'à quel niveau une réponse est pertinente ou non vis-à-vis d'une requête. Ce cadre est appelé arbre de recouvrement. Il est issu du rapprochement entre l'approche par relaxation et l'approche basée sur la distance d'édition. La relaxation concerne les descendants, lorsqu'un critère de descendance directe relaxe vers un critère de descendance indirect. L'arbre de recouvrement peut être également vu comme un cas particulier de la distance d'édition si on n'autorise qu'une seule opération élémentaire d'édition : l'ajout de nœud. Le cadre de la théorie des sous-ensembles flous a été utilisé dans deux cas : (i) d'abord pour représenter les préférences des utilisateurs, c'est-à-dire que les critères d'interrogation sont vus comme des préférences pondérées et non des contraintes strictes, (ii) ensuite pour évaluer le degré de correspondance des données avec un critère en utilisant la fonction d'appartenance. Pour mettre en œuvre et valider cette approche, l'auteur a développé et expérimenté un algorithme d'interrogation basé sur le principe de l'arbre de recouvrement. Cet algorithme construit les sous-arbres recouvrant de l'arbre de requête progressivement en parcourant l'arbre de cette dernière en post-ordre. Cet algorithme se compose de deux modules : le premier consiste à chercher pour un nœud tous les nœuds qui sont suffisamment similaires, le second concerne la validation de la structure.

Dans de Rougemont (2005) et de Rougemont et Vieilleribière (2008), on trouve une approche pour vérifier qu'un fichier XML est conforme à une DTD. Les fichiers XML sont représentés par des arbres T, et les DTD par un langage L ou un automate (et un automate est représenté par un arbre). Les auteurs utilisent ensuite des méthodes basées sur des changements des états d'automate pour estimer la distance d'édition.

Ismael Sanz et al. (2005) utilisent l'inclusion pour trouver des motifs pertinents dans la recherche de documents XML au sein d'une collection hétérogène. Ils proposent l'inclusion approximative entre les sous-arbres d'arbre de motif et l'arbre de la collection hétérogène des documents XML. L'arbre de la collection hétérogène est créé par l'ajout

d'une racine appelé db, et toutes les racines des arbres de documents deviennent des enfants de la racine db. L'inclusion approximative (par l'indexation et la similarité entre les labels) produit des sous-arbres de collection qui sont appelés des fragments. Les auteurs proposent certains critères pour l'union des fragments du même document pour créer des régions, et assigner la similarité pour ces régions. Ces régions sont considérées comme les résultats de requête d'utilisateur, et elles sont ordonnées par la similarité.

Theodore Dalamagas et al. (2004)(2006) ont utilisé une mesure de similarité basée sur la distance d'édition pour le clustering des documents XML. Ils considèrent le clustering des documents XML par structure comme le clustering des arbres de labels. Ils calculent tout d'abord la distance d'édition, qu'ils évaluent avec des algorithmes basés sur la programmation dynamique (algorithmes de Tai (1979), Zhang (1996), ...). Pour augmenter la vitesse de calcul de la distance d'édition, ils proposent des méthodes de réduction de structures, comme par exemple la réduction de nœuds de sous-arbre répété, la réduction de nœuds nichés, ... Ces réductions sont exécutées par un parcours pré-ordre d'arbre. Deuxièmement, ils proposent la similarité de structure de deux documents XML D_1, D_2 qui sont représentés par deux arbres de labels T_1, T_2 (respectivement) :

$$S(T_1, T_2) = d(T_1, T_2) / d'(T_1, T_2)$$

Où : $d(T_1, T_2)$ est la distance d'édition de deux arbres T_1 et T_2

$d'(T_1, T_2)$ est la distance d'édition de deux arbres T_1 et T_2 quand il n'y a que des opérations d'édition de suppression et d'insertion.

3 Proposition pour la recherche d'information structurée

A notre connaissance, peu de travaux utilisent directement la théorie des arbres pour la recherche d'information structurée ad-hoc (à part les travaux d'Alilaouar (2007)), c'est à dire pour calculer l'appariement approximatif entre une requête structurée et un document XML. La distance d'alignement et l'inclusion d'arbres étant des cas particuliers de la distance d'édition, nous proposons l'utilisation de la distance d'édition pour obtenir une mesure générale et flexible dans la recherche d'information structurée. Nous proposons ici d'utiliser l'algorithme de Tai (1979) pour calculer la distance d'édition de deux arbres, et nous combinons cette distance d'édition portant sur la structure avec le score portant sur le contenu. D'autres algorithmes que celui là auraient pu être choisis, mais l'algorithme de Tai, le premier à avoir été proposé, nous semble être le plus simple à mettre en œuvre dans un premier temps.

3.1 Algorithme de Tai

Dans la suite de cette section, nous utilisons les notations suivantes :

1. Une forêt/arbre n'ayant pas de nœud (forêt/arbre vide) est dénoté(e) par le symbole \emptyset .
2. $T(v)$: arbre avec nœud racine v

3. F,G: forêt¹
4. F-v : forêt obtenue après la suppression du nœud v de la forêt F
5. r_F : nœud racine de l'arbre extrême droite de la forêt F
6. F- T(r_F): forêt obtenue après la suppression entière de l'arbre extrême droite de la forêt F
7. n1n2 : forêt qui ne contient que deux nœuds n1, n2
8. cd(v) : coût de la suppression d'un nœud v
9. cs(a,b) : coût de substitution d'un nœud a par un nœud b
10. d(F,G) : distance d'édition entre les forêts F et G

L'algorithme de Tai est un algorithme récursif basé sur le lemme suivant (Tai (1979), Zhang (1996), Demaine et al. (2007)):

$$\begin{aligned}
 d(\emptyset, \emptyset) &= 0 \\
 d(F, \emptyset) &= d(F - r_F, \emptyset) + cd(r_F) \\
 d(\emptyset, G) &= d(\emptyset, G - r_G) + cd(r_G)
 \end{aligned}$$

$$d(F, G) = \min \begin{cases} d(F - r_F, G) + cd(r_F), \\ d(F, G - r_G) + cd(r_G), \\ d(T(r_F) - r_F, T(r_G) - r_G) + d(F - T(r_F), G - T(r_G)) + cs(r_F, r_G) \end{cases}$$

Où F, G sont des forêts

L'algorithme est alors le suivant:

```

Fonction d(F,G)
{
    si (F = ∅) alors
        si (G = ∅) alors renvoyer 0
        si non renvoyer d(F- rF, ∅)
    si (G = ∅) alors renvoyer d(∅, G- rG)
    a = d(F- rF, G) + cd(rF)
    b = d(F, G- rG) + cd(rG)
    c = d(T(rF) - rF, T(rG) - rG) + d(F- T(rF), G- T(rG)) +
cs(rF, rG)
    renvoyer {(a < [(b < c) ? b : c]) ? a : [(b < c) ? b : c]}
}

```

3.2 Application de l'algorithme à la recherche d'information structurée

Nous pouvons utiliser l'algorithme de Tai pour calculer la distance d'édition entre l'arbre de document (ou d'une partie de document) et l'arbre de la requête dans la recherche d'information dans des documents XML. Cette distance d'édition est combinée avec le score de contenu des nœuds feuilles descendants, obtenu avec un système de recherche

¹ Une **forêt** est un ensemble d'arbres. Un arbre peut être considéré comme une forêt d'un seul arbre. Après la suppression de la racine d'un arbre, il reste une forêt.

d'information XML classique (tel que XFIRM (Sauvagnat (2005)) par exemple) pour obtenir le score final d'un nœud ou un document². Nous distinguons quatre cas :

1. Cas 1 : Pour les requêtes simples du type $S[t_1, \dots, t_k]$ ³, on a :

$$\text{score}(n) = \lambda \cdot \sum \text{score}_{\text{nfi}}(t_1 \dots t_k) + (1 - \lambda) \cdot (|T(n)| - d(T(n), T(S))) \quad (1)$$

S : condition de structure de la requête
 $t_1 \dots t_k$: condition de contenu de la requête
n : nœud renvoyé (document ou partie de document (nœud interne n avec tous ses descendants))
T(n) : arbre avec nœud racine n
|T(n)| : nombre de nœuds dans l'arbre de racine n
T(S) : arbre de la requête
d(T(n), T(S)) : distance d'édition entre l'arbre racine n et l'arbre T(S) de la requête
 $\text{score}_{\text{nfi}}(t_1 \dots t_k)$: score d'un nœud feuille descendant du nœud n et contenant un ou plusieurs termes de la requête.
 λ : paramètre permettant d'affiner l'importance du contenu. $\lambda \in [0, 1]$

Le score d'un nœud n répondant à la requête est donc fonction du score de contenu des nœuds feuilles qu'il contient et de la distance d'édition entre l'arbre dont il est racine et l'arbre de la requête.

2. Cas 2 : Pour les requêtes du type $S1[t_1 \dots t_p]$ // ec : $S2[t_{p+1} \dots t_k]$ ⁴,

comme par exemple la requête *article[peinture] // ec: section[Dali]*⁵, le score des nœuds n2 qui satisfont cette requête est le suivant :

$$\text{score}(n2) = \alpha \cdot \sum \text{score}_{\text{nfi}}(t_{p+1} \dots t_k) + \beta \cdot \text{score}(n1) + (1 - \alpha - \beta) \cdot (|T(n2)| - d(T(n2), T(S2))) \quad (2)$$

$\text{score}_{\text{nfi}}(t_{p+1} \dots t_k)$: score d'un nœud feuille descendant du nœud n2 et contenant un ou plusieurs termes de la requête.

n1 : nœud qui satisfait la requête $S1[t_1 \dots t_p]$, n1 est ancêtre de n2, et $\text{score}(n1)$ est calculé comme le premier cas, et s'il n'existe pas de nœud n1, la partie $\beta \cdot \text{score}(n1)$ est égale à 0.

d(T(n2), T(S2)) : distance d'édition de l'arbre ayant pour racine n2 et du sous-arbre de la requête ayant pour racine S2

² Nous rappelons que dans un document XML, les informations de contenu sont situées au niveau des nœuds feuilles.

³ On cherche une structure S qui vérifie la condition de contenu $t_1 \dots t_k$.

⁴ On cherche une structure S2 (structure cible) qui vérifie la condition de contenu $t_{p+1} \dots t_k$, contenue dans une structure S1 (structure support) qui vérifie la condition de contenu $t_1 \dots t_p$.

⁵ Cette requête signifie que l'on cherche une section qui parle de « Dali » (structure cible) contenue dans un article qui parle de « peinture » (structure support).

$\alpha, \beta \in [0,1]$ sont des paramètres permettant d'affiner respectivement l'importance donnée au contenu et au nœud ancêtre.

Le score d'un nœud $n2$ répondant à la requête est donc fonction du score de contenu des nœuds feuilles qu'il contient, de la distance d'édition entre l'arbre dont il est racine et le sous-arbre de la requête portant sur l'élément cible, et du score d'un nœud $n1$ étant son ancêtre et répondant à la première partie de la requête.

3. Cas 3 : Pour les requêtes du type $S1[t_1...t_p] // S2[t_{p+1}...t_k]^6$,

le score des nœuds $n1$ qui satisfont cette requête est le suivant :

$$\text{score}(n1) = \alpha \cdot \sum \text{score}_{\text{nfi}}(t_1...t_p) + \beta \cdot \sum \text{score}(n2) + (1 - \alpha - \beta) \cdot (|T(n1)| - d(T(n1), T(S1))) \quad (3)$$

$n2$: nœud qui satisfait la requête $S2[t_{p+1}...t_k]$, $n2$ est descendant de $n1$, et $\text{score}(n2)$ est calculé comme le premier cas.

$d(T(n1), T(S1))$: distance d'édition de l'arbre ayant pour racine $n1$ et l'arbre de la requête $T(S1)$.

$\alpha, \beta \in [0,1]$: paramètres permettant d'affiner respectivement l'importance donnée au contenu et aux nœuds internes descendants.

Le score d'un nœud $n1$ répondant à la requête est donc fonction du score de contenu des nœuds feuilles qu'il contient, de la distance d'édition entre l'arbre dont il est racine et le sous-arbre de la requête portant sur l'élément cible, et du score d'un nœud $n2$ étant son descendant et répondant à la seconde partie de la requête.

4. Cas 4 : Pour les requêtes du type $S1[t_1...t_p] // ec : S2[t_{p+1}...t_q] // S3[t_{q+1}...t_k]^7$,

le score des nœuds $n2$ qui satisfont cette requête est le suivant :

$$\text{score}(n2) = \alpha \cdot \sum \text{score}_{\text{nfi}}(t_{p+1}...t_q) + \beta \cdot \sum \text{score}(n3) + \gamma \cdot \text{score}(n1) + (1 - \alpha - \beta - \gamma) \cdot (|T(n2)| - d(T(n2), T(S2))) \quad (4)$$

$n1$: nœud qui satisfait la requête $S1[t_1...t_p]$, $n1$ est ancêtre de $n2$, et $\text{score}(n1)$ est calculé comme le premier cas.

$n3$: nœud qui satisfait la requête $S3[t_{q+1}...t_k]$, $n3$ est descendant de $n2$, et $\text{score}(n3)$ est calculé comme le premier cas.

$d(T(n2), T(S2))$: distance d'édition de l'arbre ayant pour racine $n2$ et du sous-arbre de requête ayant pour racine $S2$

⁶ On cherche une structure $S1$ (structure cible) qui vérifie la condition de contenu $t_1...t_p$, et contenant une structure $S2$ (structure support) qui vérifie la condition de contenu $t_{p+1}...t_k$.

⁷ On cherche une structure $S2$ (structure cible) qui vérifie la condition de contenu $t_{p+1}...t_q$, contenue dans une structure support $S1$ vérifiant $t_1...t_p$ et contenant une structure $S3$ vérifiant $t_{q+1}...t_k$.

$\alpha, \beta, \gamma \in [0,1]$: paramètres permettant d'affiner respectivement l'importance donnée au contenu, aux nœuds internes descendants, et au nœud ancêtre.

Le score d'un nœud n_2 répondant à la requête est donc fonction du score de contenu des nœuds feuilles qu'il contient, de la distance d'édition entre l'arbre dont il est racine et le sous-arbre de la requête portant sur l'élément cible, du score d'un nœud n_3 étant son ancêtre et répondant à la première partie de la requête et enfin du score d'un nœud n_1 étant son descendant et répondant à la troisième partie de la requête.

Les types de requêtes énoncés ici sont représentatifs de la majorité des requêtes structurées portant sur des documents structurés. Pour plus d'information sur ces requêtes, on peut se référer à Trotman et Sigurbjornsson (2005) et Sauvagnat (2005).

4 Illustration sur un exemple

Nous présentons dans cette section un exemple de calcul des scores de pertinence pour le document XML de la figure 1 (A) et la requête de la figure 1 (B) : *article[peinture] // ec: section[Dali]* (recherche de sections qui parlent de « Dali », dans un article qui parle de « peinture »), en utilisant l'algorithme combinaison de Tai et le score des nœuds feuilles descendants calculé sur le contenu avec un système de recherche d'information XML classique.

Pour calculer la distance d'édition, le coût de suppression d'un nœud v ($cd(v)$) et le coût de substitution d'un nœud a par un nœud b ($cs(a,b)$) sont proposés par exemple comme suit :

$$cd(v) = 1$$

$$cs(a,b) = 1 - \text{similarité}(a,b)$$

Lorsque $cd(v) = 1$, $d(F, \emptyset) = d(\emptyset, F) =$ nombre de nœuds de F .

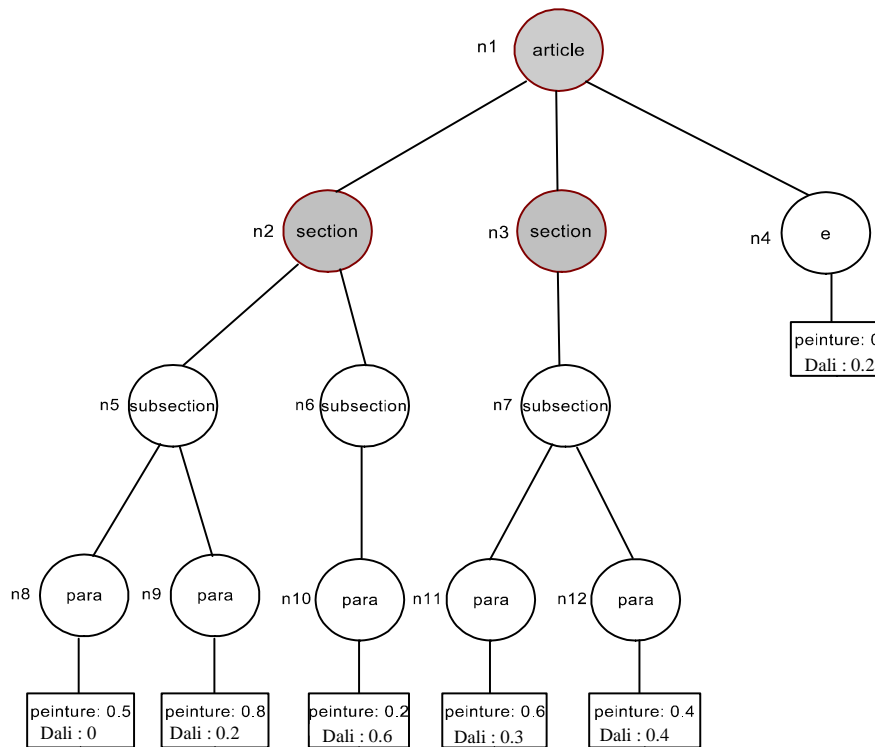


FIG. 1 (A) – *Arbre du document.*

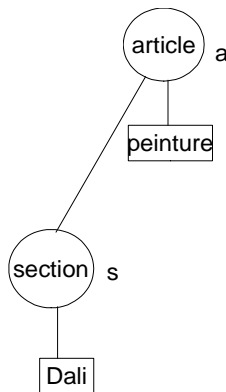


FIG. 1 (B) – *Arbre de la requête.*

Le tableau 1 donne les similarités possibles entre les nœuds du document et les nœuds de la requête. Ces similarités sont évaluées manuellement, en fonction de la sémantique des balises des nœuds concernés. Par exemple, un nœud *section* et un nœud *subsection*

ont une similarité de 1/2, (ils ont des sémantiques relativement proches) alors qu'un nœud *section* et un nœud *para* ont une similarité de 1/4 (ils ont des sémantiques plus éloignées).

Le coût de substitution $cs(a,b)$ est égal à $1 - \text{similarité}(a,b)$. Par exemple $cs(a,n1) = 1 - 1 = 0$; $cs(s,n5) = 1 - 1/2 = 1/2$.

	n1 (article)	n2,n3 (section)	n4 (e)	N5,n6,n7 (subsection)	n8=>n12 (para)
a(article)	1	1/2	0	0	0
s(section)	1/2	1	0	1/2	1/4

TAB. 1 – Similarité entre les nœuds

Au début, deux requêtes structurées *article[peinture]* et *section[Dali]* sont traitées par le système de recherche XML classique . Les résultats de ces requêtes sont respectivement le nœud n1 (article) et les deux nœuds section (n2,n3).

Ensuite, trois distances d'édition : la distance $d(T(n1),T(a))$ de l'arbre $T(n1)$ avec nœud racine n1 et l'arbre de requête $T(a)$, la distance $d(T(n2), T(s))$ de l'arbre avec nœud racine n2 et le sous-arbre de requête $T(s)$ avec nœud racine s, et la distance $d(T(n3),T(s))$ de l'arbre racine n3 et le sous-arbre $T(s)$ de la requête avec racine s sont calculées par l'algorithme de Tai. Les étapes du calcul de ces distances sont détaillées en annexe 1.

Le score de nœud article n1 est calculé ainsi:

$$\text{score}(n1) = \lambda \cdot \sum_{nf_i \in \text{descendants}(n1)} \text{score}_{nf_i}(\text{peinture}) + (1 - \lambda) \cdot (|T(n1)| - d(T(n1),T(a)))$$

où : $\text{descendants}(n1)$ est l'ensemble des nœuds feuilles descendants de n1.

$\text{score}_{nf_i}(\text{peinture})$ est le score de pertinence du nœud feuille nf_i descendant de n1 qui contient le mot clé « *peinture* »

$$d(T(n1),T(a)) = 10 \text{ et } |T(n1)| = 12$$

Enfin, les résultats de la requête sont les nœuds section n2 et n3, avec le score final

$$\text{score}(n2) = \alpha \times \sum_{nf_i \in \text{descendants}(n2)} \text{score}_{nf_i}(\text{Dali}) + \beta \times \text{score}(n1) + (1 - \alpha - \beta) \times (|T(n2)| - d(T(n2),T(s)))$$

$$\text{score}(n3) = \alpha \times \sum_{nf_i \in \text{descendants}(n3)} \text{score}_{nf_i}(\text{Dali}) + \beta \times \text{score}(n1) + (1 - \alpha - \beta) \times (|T(n3)| - d(T(n3),T(s)))$$

où : $\text{descendants}(n2)$ est l'ensemble des nœuds feuilles descendants de n2.

$\text{descendants}(n3)$ est l'ensemble des nœuds feuilles descendants de n3.

$score_{nf_i}(Dali)$ est le score de pertinence du nœud feuille nf_i qui contient le mot clé « *Dali* ».

$$d(T(n2),T(s)) = 5, d(T(n3),T(s)) = 3, |T(n2)| = 6, \text{ et } |T(n3)| = 4$$

5 Conclusion et Perspectives

Dans cet article, nous avons proposé une mesure de similarité de structure entre des documents XML. Cette mesure est basée sur la distance d'édition des arbres qui représentent ces documents, en utilisant l'algorithme de Tai (1979). La similarité entre des documents XML et la requête est la combinaison de la similarité de structure et la similarité de contenu (en utilisant la similarité de contenu d'un moteur de recherche XML classique).

Nos propositions sont en cours d'évaluation sur les collections d'INEX, tâche Ad-hoc. INEX (*Initiative for the Evaluation of XML Retrieval*) est la campagne d'évaluation traditionnellement utilisée pour évaluer les systèmes de recherche d'information structurée.

Notre approche de recherche ad-hoc peut cependant être améliorée selon plusieurs points :

- Concernant la vitesse de calcul de la distance d'édition : (i) des algorithmes plus rapides comme par exemple l'algorithme de Demaine et al. (2007) ou encore des algorithmes tournant sur des arbres de faible hauteur peuvent être utilisés; (ii) des contraintes issues de la DTD peuvent être utilisées pour réduire les calculs et des limitations sur des opérations d'édition peuvent également être mises en œuvre.
- Une mesure de similarité flexible peut être obtenue en utilisant des opérations d'édition floues.
- La combinaison entre la similarité de contenu et la similarité de structure doit être approfondie.

D'autre part, nous envisageons de proposer des méthodes basées sur les graphes pour résoudre d'autres problèmes de la recherche d'information structurée, comme par exemple les problèmes d'hétérogénéité de corpus et de classification de documents XML. Ces problématiques et les solutions associées pourront être évaluées sur les tâches hétérogènes et XML Mining de la campagne d'évaluation INEX.

6 Bibliographie

- Alilaouar A. (2007). *Contribution à l'interrogation flexible de données semi-structurées*. Thèse de l'Université Paul Sabatier 2007.
- Bunke H. (1998). *Error-Tolerant Graph Matching: A Formal Framework and Algorithms*. Lecture notes in computer science, 1998 – Springer.
- Bunke H., Shearer K. (1998). *A graph distance metric based on the maximal common sub-graph*. Pattern Recognition Letters 19(1998)255–259.
- Bunke H. (2000). *Recent Developments in Graph Matching*, IEEE 2000.

- Chen W. (1998). *More Efficient Algorithm for Ordered Tree Inclusion*. JOURNAL OF ALGORITHMS, Academic Press 1998.
- Dalamagas T., Cheng T., Winkel K., and Sellis T. (2004). *Clustering XML Documents Using Structural Summaries*. W. Lindner et al. (Eds.): EDBT 2004 Workshops, LNCS 3268, pp. 547–556, 2004.
- Dalamagas T., Cheng T., Winkel K., and Sellis T. (2006). A methodology for clustering XML documents by structure. *Information Systems* 31.
- Demaine E. D., Mozes S., Rossmao B., et Weimann O. (2007). *An Optimal Decomposition Algorithm for Tree Edit Distance*. ACM Journal Name, Vol. TBD, No. TBD, TBD 20TBD (2007).
- Fuhr N. Kamps J., Lalmas M., Trotman A. (2008). *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007*, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers, Springer, Lecture Notes in Computer Science, volume 4862, 2008.
- Geva S., Kamps J., Trotman A. (2009) *Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008*, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers, Springer, Lecture Notes in Computer Science, volume 5631, 2009.
- Hidovi D., Pelillo M. (2004) . *Metrics for Attributed Graphs Based on the Maximal Similarity Common Subgraph*. International Journal of Pattern Recognition and Artificial, 7/2004.
- Jiang T., Wang L. , Zhang K. (1995). *Alignment of trees - an alternative to tree edit*. Theoretical Computer Science 143, 1995.
- Klein P. N. (1998). *Computing the Edit-Distance Between Unrooted Ordered Trees*. Springer-Verlag Berlin Heidelberg 1998.
- Lee M. D. (2009). *Graphes et appariement sémantique de documents XML*. Rapport de Master 2 Recherche, Université Paul Sabatier, 2009.
- Manning C. D., Raghavan P., et Schütze H (2008). *Introduction to Information Retrieval*. Cambridge University Press 2008
- Pinel-Sauvagnat K., Boughanem M (2006). *Propositions pour la pondération des termes et l'évaluation de la pertinence des éléments en recherche d'information structurée*. Revue I3 - Information Interaction Intelligence - Volume 6, n°2.
- de Rougemont M. (2005). *The correction of XML data*. Lecture notes in computer science, 2005.
- de Rougemont M., Vieillerivière A.. *Approximate Schemas, Source-Consistency and Query Answering*. Journal of Intelligent Information Systems, 2008 – Springer.

- Sauvagnat K. (2005). *Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés*. Thèse de l' Université Paul Sabatier de Toulouse 2005.
- Shasha D. , Zhang K. (1997). *Approximate tree pattern matching*. In Pattern Matching Algorithms, 1997.
- Sanz I., Mesiti M., Guerrini G., and Berlanga Llavori R. (2005). *Approximate Subtree Identification in Heterogeneous XML Documents Collections*. Lecture notes in computer science, 2005.
- Tai K (1979). *The tree-to-tree correction problem*, J. ACM 26 (3), p. 422-433.
- Trotman A., Sigurbjornsson B. (2005). *NEXI, Now and Next*. INEX 2004, Springer-Verlag Berlin Heidelberg 2005.
- Wallis W.D. , Shoubridge P., Kraetz M., Ray D.. *Graph distances using graph union*. Pattern Recognition Letters 22 (2001).
- Zhang K. (1996). *A Constrained Edit Distance Between Unordered Labeled Trees*. Springer-Verlag, Algorithmica 1996.

Annexe 1

Étapes de calcul de la distance d'édition $d(T(n1), T(a))$:

$$\begin{aligned}
 d(T(n1), T(a)) &= \min \{ d(T(n1)-n1, T(a)) + cd(n1), \\
 &\quad d(T(n1), s) + cd(a), \\
 &\quad d(T(n1)-n1, s) + d(\emptyset, \emptyset) + cs(n1, a) \} \\
 d(T(n1)-n1, T(a)) &= \min \{ d(T(n1)-n1-n4, T(a)) + cd(n4), \\
 &\quad d(T(n1)-n1, s) + cd(a), \\
 &\quad d(\emptyset, s) + d(T(n1)-n1-n4, \emptyset) + cs(n4, a) \} \\
 d(T(n1), s) &= \min \{ d(T(n1)-n1, s) + cd(n1), \\
 &\quad d(T(n1), \emptyset) + cd(s), \\
 &\quad d(T(n1)-n1, \emptyset) + d(\emptyset, \emptyset) + cs(n1, s) \} \\
 d(T(n1)-n1, s) &= \min \{ d(T(n1)-n1-n4, s) + cd(n4), \\
 &\quad d(T(n1)-n1, \emptyset) + cd(s), \\
 &\quad d(\emptyset, \emptyset) + d(T(n1)-n1-n4, \emptyset) + cs(n4, s) \} \\
 d(T(n1)-n1-n4, T(a)) &= \min \{ d(T(n1)-n1-n4-n3, T(a)) + cd(n3), \\
 &\quad d(T(n1)-n1-n4, s) + cd(a), \\
 &\quad d(T(n1), s) + d(T(n2), \emptyset) + cs(n3, a) \} \\
 d(T(n1)-n1-n4, s) &= \min \{ d(T(n1)-n1-n4-n3, s) + cd(n3), \\
 &\quad d(T(n1)-n1-n4, \emptyset) + cd(s), \\
 &\quad d(T(n1), \emptyset) + d(T(n2), \emptyset) + cs(n3, s) \} \\
 d(T(n1)-n1-n4-n3, T(a)) &= \min \{ d(T(n1)-n1-n4-n3-n7, T(a)) + cd(n7), \\
 &\quad d(T(n1)-n1-n4-n3, s) + cd(a), \\
 &\quad d(n1 \ n12, s) + d(T(n2), \emptyset) + cs(n7, a) \} \\
 d(T(n1), s) &= \min \{ d(n1 \ n12, s) + cd(n7),
 \end{aligned}$$

$$\begin{aligned}
& d(T(n7), n) + cd(s), \\
& d(n11n12, \emptyset) + d(\emptyset, \emptyset) + cs(n7, s) \} \\
d(T(n1)-n1-n4-n3, s) = & \min \{ d(T(n1)-n1-n4-n3-n7, s) + cd(n7), \\
& d(T(n1)-n1-n4-n3, \emptyset) + cd(s), \\
& d(n11n12, \emptyset) + d(T(n2), \emptyset) + cs(n7, s) \} \\
d(T(n1)-n1-n4-n3-n7, T(a)) = & \min \{ d(T(n1)-n1-n4-n3-n7-n12, T(a)) + cd(n12), \\
& d(T(n1)-n1-n4-n3-n7, s) + cd(a), \\
& d(\emptyset, s) + d(T(n2) + n11, \emptyset) + cs(n12, a) \} \\
d(n11n12, s) = & \min \{ d(n11, s) + cd(n12), \\
& d(n11n12, \emptyset) + cd(s), \\
& d(\emptyset, \emptyset) + d(n11, \emptyset) + cs(n12, s) \} \\
d(T(n1)-n1-n4-n3-n7, s) = & \min \{ d(T(n1)-n1-n4-n3-n7-n12, s) + cd(n12), \\
& d(T(n1)-n1-n4-n3-n7, \emptyset) + cd(s), \\
& d(\emptyset, n) + d(T(n2) + n11, \emptyset) + cs(n12, s) \} \\
d(T(n1)-n1-n4-n3-n7-n12, T(a)) = & \min \{ d(T(n2), T(a)) + cd(n11), \\
& d(T(n1)-n1-n4-n3-n7-n12, s) + cd(a), \\
& d(\emptyset, s) + d(T(n2), \emptyset) + cs(n11, a) \} \\
d(n11, s) = & \min \{ d(\emptyset, s) + cd(n11), \\
& d(n11, \emptyset) + cd(s), \\
& d(\emptyset, \emptyset) + d(\emptyset, \emptyset) + cs(n11, s) \} \\
d(T(n1)-n1-n4-n3-n7-n12, s) = & \min \{ d(T(n2), s) + cd(n11), \\
& d(T(n1)-n1-n4-n3-n7-n12, \emptyset) + cd(s), \\
& d(\emptyset, \emptyset) + d(T(n2), \emptyset) + cs(n11, s) \} \\
d(T(n2), T(a)) = & \min \{ d(T(n2)-n2, T(a)) + cd(n2), \\
& d(T(n2), s) + cd(a), \\
& d(T(n2)-n2, s) + d(\emptyset, \emptyset) + cs(n2, a) \} \\
d(T(n2), s) = & \min \{ d(T(n2)-n2, s) + cd(n2), \\
& d(T(n2), \emptyset) + cd(s), \\
& d(T(n2)-n2, \emptyset) + d(\emptyset, \emptyset) + cs(n2, s) \} \\
d(T(n2)-n2, T(a)) = & \min \{ d(T(n2)-n2-n6, T(a)) + cd(n6), \\
& d(T(n2)-n2, s) + cd(a), \\
& d(n10, s) + d(T(n5), \emptyset) + cs(n6, a) \} \\
d(T(n2)-n2, s) = & \min \{ d(T(n2)-n2-n6, s) + cd(n6), \\
& d(T(n2)-n2, \emptyset) + cd(s), \\
& d(n10, \emptyset) + d(T(n5), \emptyset) + cs(n6, s) \} \\
d(T(n2)-n2-n6, T(a)) = & \min \{ d(T(n5), T(a)) + cd(n10), \\
& d(T(n2)-n2-n6, s) + cd(a), \\
& d(\emptyset, s) + d(T(n5), \emptyset) + cs(n10, a) \} \\
d(n10, s) = & \min \{ d(\emptyset, s) + cd(n10), \\
& d(n10, \emptyset) + cd(s), \\
& d(\emptyset, \emptyset) + d(\emptyset, \emptyset) + cs(n10, s) \} \\
d(T(n2)-n2-n6, s) = & \min \{ d(T(n5), s) + cd(n10), \\
& d(T(n2)-n2-n6, \emptyset) + cd(s), \\
& d(\emptyset, \emptyset) + d(T(n5), \emptyset) + cs(n10, s) \} \\
d(T(n5), T(a)) = & \min \{ d(n8n9, T(a)) + cd(n5),
\end{aligned}$$

$$\begin{aligned}
& d(T(n5),s) + cd(a), \\
& d(n8n9,s) + d(\emptyset,\emptyset) + cs(n5,a) \} \\
d(T(n5),s) &= \min \{ d(n8n9,s) + cd(n5), \\
& d(T(n5),\emptyset) + cd(s), \\
& d(n8n9,\emptyset) + d(\emptyset,\emptyset) + cs(n5,s) \} \\
d(n8n9,T(a)) &= \min \{ d(n8,T(a)) + cd(n9), \\
& d(n8n9,s) + cd(a), \\
& d(\emptyset,s) + d(n8,\emptyset) + cs(n9,a) \} \\
d(n8n9,s) &= \min \{ d(n8,s) + cd(n9), \\
& d(n8n9,\emptyset) + cd(s), \\
& d(\emptyset,\emptyset) + d(n8,\emptyset) + cs(n9,s) \} \\
d(n8,T(a)) &= \min \{ d(\emptyset,T(a)) + cd(n8), \\
& d(n8,s) + cd(a), \\
& d(\emptyset,s) + d(\emptyset,\emptyset) + ds(n8,a) \} \\
d(n8,s) &= \min \{ d(\emptyset,s) + cd(n8), \\
& d(n8,\emptyset) + cd(s), \\
& d(\emptyset,\emptyset) + cs(n8,s) \} \\
(& d(n8,s) = \min \{ 1 + 1, 1 + 1, 1 - 1/4 \} = 3/4 \\
& d(n8,T(a)) = \min \{ 2 + 1, 3/4 + 1, 1 + 1 \} = 7/4 \\
& d(n8n9,s) = \min \{ 7/4 + 1, 2 + 1, 1 + 3/4 \} = 7/4 \\
& d(n8n9,T(a)) = \min \{ 1, 7/4 (10.2) + 1, 1 + 1 + 1 \} = 11/4 \\
& d(T(n5),s) = \min \{ 7/4 + 1, 3 + 1, 2 + 0 + 1/2 \} = 5/2 \\
& d(T(n5),T(a)) = \min \{ 11/4 + 1, 5/2 + 1, 7/4 + 0 + 1 \} = 11/4 \\
& d(T(n2)-n2-n6,s) = \min \{ 5/2 + 1, 4 + 1, 3 + 3/4 \} = 7/2 \\
& d(n10,s) = \min \{ 1 + 1, 1 + 1, 3/4 \} = 3/4 \\
& d(T(n2)-n2-n6,T(a)) = \min \{ 11/4 + 1, 3.75, 7/2 + 1, 4.5, 1 + 3 + 1 \} = 3.75 \\
& d(T(n2)-n2,s) = \min \{ 7/2 + 1, 5 + 1, 1 + 3 + 1/2 \} = 4.5 \\
& d(T(n2)-n2,T(a)) = \min \{ 3.75 + 1, 4.5 + 1, 3/4 + 3 + 1 \} = 4.75 \\
& d(T(n2),s) = \min \{ 4.5 + 1, 6 + 1, 5 + 0 + 0 \} = 5 \\
& d(T(n2),T(a)) = \min \{ 4.75 + 1, 5 + 1, 4.5 + 1/2 \} = 5 \\
& d(T(n1)-n1-n4-n3-n7-n12,s) = \min \{ 5 + 1, 7 + 1, 0 + 6 + 3/4 \} = 6 \\
& d(n11,s) = \min \{ 1 + 1, 1 + 1, 0 + 3/4 \} = 0.75 \\
& d(T(n1)-n1-n4-n3-n7-n12,T(a)) = \min \{ 5 + 1, 6 + 1, 1 + 6 + 1 \} = 6 \\
& d(T(n1)-n1-n4-n3-n7,s) = \min \{ 6 + 1, 8 + 1, 0 + 7 + 3/4 \} = 7 \\
& d(n11n12,s) = \min \{ 0.75 + 1, 2 + 1, 0 + 1 + 0.75 \} = 1.75 \\
& d(T(n1)-n1-n4-n3-n7,T(a)) = \min \{ 6 + 1, 7 + 1, 1 + 7 + 1 \} = 7 \\
& d(T(n1)-n1-n4-n3,s) = \min \{ 7 + 1, 9 + 1, 2 + 6 + 1/2 \} = 8 \\
& d(T(n7),s) = \min \{ 1.75 + 1, 3 + 1, 2 + 0 + 1/2 \} = 2.5 \\
& d(T(n1)-n1-n4-n3,T(a)) = \min \{ 7 + 1, 8 + 11.75 + 6 + 1 \} = 8 \\
& d(T(n1)-n1-n4,s) = \min \{ 8 + 1, 10 + 1, 3 + 6 + 0 \} = 9 \\
& d(T(n1)-n1-n4,T(a)) = \min \{ 8 + 1, 9 + 1, 2.5 + 6 + 1/2 \} = 9 \\
& d(T(n1)-n1,s) = \min \{ 9 + 1, 11 + 1, 0 + 10 + 1 \} = 10 \\
& d(T(n1),s) = \min \{ 10 + 1, 12 + 1, 11 + 0 + 1/2 \} = 11 \\
& d(T(n1)-n1,T(a)) = \min \{ 9 + 1, 10 + 1, 1 + 10 + 1 \} = 10 \\
& d(T(n1),T(a)) = \min \{ 10 + 1, 11 + 1, 10 + 0 + 0 \} = 10)
\end{aligned}$$

$$\underline{d(T(n1),T(a))} = 10$$

Étapes de calcul de la distance d'édition $d(T(n2),T(s))$

$$\begin{aligned} d(T(n2),T(s)) &= d(T(n2),s) = \min \{ d(T(n2)-n2,s) + cd(n2), \\ &\quad d(T(n2),\emptyset) + cd(s), \\ &\quad d(T(n2)-n2,\emptyset) + d(\emptyset,\emptyset) + cs(n2,s) \} \\ d(T(n2)-n2,s) &= \min \{ d(T(n2)-n2-n6,s) + cd(n6), \\ &\quad d(T(n2)-n2,\emptyset) + cd(s), \\ &\quad d(n10,\emptyset) + d(T(n5),\emptyset) + cs(n6,s) \} \\ d(T(n2)-n2-n6,s) &= \min \{ d(T(n5),s) + cd(n10), \\ &\quad d(T(n2)-n2-n6,\emptyset) + cd(s), \\ &\quad d(\emptyset,\emptyset) + d(T(n5),\emptyset) + cs(n10,s) \} \\ d(T(n5),s) &= \min \{ d(n8n9,s) + cd(n5), \\ &\quad d(T(n5),\emptyset) + cd(s), \\ &\quad d(n8n9,\emptyset) + d(\emptyset,\emptyset) + cs(n5,s) \} \\ d(n8n9,s) &= \min \{ d(n8,s) + cd(n9), \\ &\quad d(n8n9,\emptyset) + cd(s), \\ &\quad d(\emptyset,\emptyset) + d(n8,\emptyset) + cs(n9,s) \} \\ d(n8,s) &= \min \{ d(\emptyset,s) + cd(n8), \\ &\quad d(n8,\emptyset) + cd(s), \\ &\quad d(\emptyset,\emptyset) + d(\emptyset,\emptyset) + cs(n8,s) \} \\ (\quad d(n8,s) &= \min \{ 1 + 1, 1 + 1, \quad 0 + 0 + 0.75 \} = 0.75 \\ d(n8n9,s) &= \min \{ 0.75 + 1, 2 + 1, 0 + 1 + 0.75 \} = 1.75 \\ d(T(n5),s) &= \min \{ 1.75 + 1, 3 + 1, 2 + 0 + 1/2 \} = 2.5 \\ d(T(n2)-n2-n6,s) &= \min \{ 2.5 + 1, 4 + 1, 0 + 3 + 0.75 \} = 3.5 \\ d(T(n2)-n2,s) &= \min \{ 3.5 + 1, 5 + 1, 1 + 3 + 1/2 \} = 4.5 \\ d(T(n2),s) &= \min \{ 4.5 + 1, 6 + 1, 5 + 0 + 0 \} = 5 \end{aligned}$$

$$\underline{d(T(n2),T(s))} = 5$$

Étapes de calcul de la distance d'édition $d(T(n3),T(s))$

$$\begin{aligned} d(T(n3),T(s)) &= d(T(n3),s) = \min \{ d(T(n7),s) + cd(n3), \\ &\quad d(T(n3),\emptyset) + cd(s), \\ &\quad d(T(n7),\emptyset) + d(\emptyset,\emptyset) + cs(n3,s) \} \\ d(T(n7),s) &= \min \{ d(n11n12,s) + cd(n7), \\ &\quad d(T(n7),\emptyset) + cd(s), \\ &\quad d(n11n12,\emptyset) + d(\emptyset,\emptyset) + cs(n7,s) \} \\ d(n11n12,s) &= \min \{ d(n11,s) + cd(n12), \\ &\quad d(n11n12,\emptyset) + cd(s), \\ &\quad d(\emptyset,\emptyset) + d(n11,\emptyset) + cs(n12,s) \} \\ d(n11,s) &= \min \{ d(\emptyset,s) + cd(n11), \end{aligned}$$

$$\begin{aligned}
 & d(n11, \emptyset) + cd(s), \\
 & d(\emptyset, \emptyset) + d(\emptyset, \emptyset) + cs(n11, s) \\
 (& d(n11, s) = \min \{ 1 + 1, 1 + 1, 0 + 0.75 \} = 0.75 \\
 & d(n11n12, s) = \min \{ 0.75 + 1, 2 + 1, 0 + 1 + 0.75 \} = 1.75 \\
 & d(T(n7), s) = \min \{ 1.75 + 1, 3 + 1, 2 + 0 + 1/2 \} = 2.5 \\
 & d(T(n3), s) = \min \{ 2.5 + 1, 4 + 1, 3 + 0 + 0 \} = 3)
 \end{aligned}$$

$$\underline{d(T(n3), T(s))} = 3$$

Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-08-CORD-009.